BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

What is Claimed is:

1. A method of identifying an errant process in a computer system, the method comprising:

detecting an error;

storing a physical address of an errant process that caused the error;

storing an execution instruction pointer (IP) in an interruption instruction pointer (IIP);

determining a first virtual address from an operating system mapping table;

determining a second virtual address from a translation look-aside buffer; and

identifying the errant process, if the physical address and the second virtual address match the physical address and the first virtual address.

2. The method of claim 1, further comprising:

determining whether the physical address of the physical memory location is known;

determining in which code section the errant process is located, if the physical memory location is known;

resetting the processor, if the physical memory location is in one of a critical section and an unknown section of the code; and

terminating the errant process based on a level of sharing of the physical memory location and whether the IIP associated with the errant process is precise, if the physical memory location is in a non-critical section of the code.

3. The method of claim 2, said terminating the errant process based on a level of sharing of the physical memory location and whether the IIP associated with the errant process is precise, comprising:

determining whether the physical memory location is one of a global, a shared and a private physical memory location;

determining whether the IIP associated with the errant process is precise;

terminating the errant process, if the physical memory location is global and the IIP is precise; and

resetting the processor, if the physical memory location is global and the IIP is imprecise.

4. The method of claim 2, said terminating the errant process based on a level of sharing of the physical memory location and whether the IIP associated with the errant process is precise, comprises:

determining whether the physical memory location is one of a global, a shared and a private physical memory location;

determining whether the IIP associated with the errant process is precise;

terminating the errant process, if the physical memory location is shared and the IIP is precise; and

determining whether the physical address can be used to indicate the errant process, if the physical memory location is shared and the IIP is imprecise; and

terminating the errant process, if the physical address can be used to indicate the errant process; otherwise, resetting the processor.

5. The method of claim 2, said terminating the errant process based on a level of sharing of the physical memory location and whether the IIP associated with the errant process is precise, comprises:

determining whether the physical memory location is one of a global, a shared and a private physical memory location;

determining whether the IIP associated with the errant process is precise;

terminating the errant process specified by the precise IIP, if the physical memory location is private and the IIP is precise; and

determining whether the physical address can be used to indicate the errant process, if the physical memory location is private and the IIP is imprecise;

terminating the errant process, if the physical address can be used to indicate the errant process; otherwise, resetting the processor.

6. The method of claim 1, said storing the physical address of the errant process that caused the error comprising:

storing the physical address of the errant process in a memory register.

7. The method of claim 1, said determining the first virtual address from the operating system mapping table comprising:

finding a physical address entry in the operating system mapping table that matches the physical address; and

reading the first virtual address from the matching physical address entry, if the matching physical address entry is found.

8. The method of claim 1, said determining the second virtual address from the translation look-aside buffer comprising:

finding a physical address entry in the translation look-aside buffer that matches the physical address; and

reading the second virtual address from the matching physical address entry, if the matching physical address entry is found.

9. The method of claim 1 said execution instruction pointer comprises:

an instruction pointer that existed at the time the error is detected.

10. The method of claim 1, said operating system mapping table comprises one of:

a buffer; and

a cache array.

11. A machine-readable medium having stored thereon a plurality of executable instructions, the plurality of instructions comprising instructions to:

detect an error;

store a physical address of an errant process that caused the error;

store an execution instruction pointer (IP) in an interruption instruction pointer (IIP);

determine a first virtual address from an operating system mapping table;

determine a second virtual address from a translation look-aside buffer;

identify the errant process, if the physical address and the second virtual address match the physical address and the first virtual address.

12. The machine-readable medium of claim 11, further comprising instructions to:

determine whether the physical address of the physical memory location is known;

determine in which code section the errant process is located, if the physical memory location is known;

reset the processor, if the physical memory location is in one of a critical section and an unknown section of the code; and

terminate the errant process based on a level of sharing of the physical memory location and whether the IIP associated with the errant process is precise, if the physical memory location is in a non-critical section of the code.

13. The machine-readable medium of claim 12, said terminate the errant process based on a level of sharing of the physical memory location and whether the IIP associated with the errant process is precise instruction comprising instructions to:

determine whether the physical memory location is one of a global, a shared and a private physical memory location;

determine whether the IIP associated with the errant process is precise;

terminate the errant process, if the physical memory location is global and the IIP is precise; and

reset the processor, if the physical memory location is global and the IIP is imprecise.

14. The machine-readable medium of claim 12, said terminate the errant process based on a level of sharing of the physical memory location and whether the IIP associated with the errant process is precise instruction comprising instructions to:

determine whether the physical memory location is one of a global, a shared and a private physical memory location;

determine whether the IIP associated with the errant process is precise;

terminate the errant process, if the physical memory location is shared and the IIP is precise; and

determine whether the physical address can be used to indicate the errant process, if the physical memory location is shared and the IIP is imprecise; and

terminate the errant process, if the physical address can be used to indicate the errant process; otherwise, reset the processor.

15. The machine-readable medium of claim 12, said terminate the errant process based on a level of sharing of the physical memory location and whether the IIP associated with the errant process is precise instruction comprising instructions to:

determine whether the physical memory location is one of a global, a shared and a private physical memory location;

determine whether the IIP associated with the errant process is precise;

terminate the errant process specified by the precise IIP, if the physical memory location is private and the IIP is precise; and

determine whether the physical address can be used to indicate the errant process, if the physical memory location is private and the IIP is imprecise;

terminate the errant process, if the physical address can be used to indicate the errant process; otherwise, reset the processor.

16. The machine-readable medium of claim 11, said store the physical address of the errant process that caused the error instruction comprising an instruction to:

store the physical address of the errant process in a memory register.

17. The machine-readable medium of claim 11, said determining the first virtual address from the operating system mapping table instruction comprising instructions to:

find a physical address entry in the operating system mapping table that matches the physical address; and

read the first virtual address from the matching physical address entry, if the matching physical address entry is found.

18. The machine-readable medium of claim 11, said determine the second virtual address from the translation look-aside buffer instruction comprising instructions to:

find a physical address entry in the translation look-aside buffer that matches the physical address; and

read the second virtual address from the matching physical address entry, if the matching physical address entry is found.

19. The machine-readable medium of claim 11, said store the execution instruction pointer in the interruption instruction pointer instruction comprising an instruction to:

store an instruction pointer that existed at the time the error is detected in the interruption instruction pointer.

20. A computer system, comprising:

a processor;

a system memory coupled to the processor; and

a machine-readable medium coupled to the processor, said machine-readable medium having stored thereon a plurality of executable instructions, the plurality of instructions comprising instructions to:

detect an error;

store a physical address of an errant process that caused the error;

store an execution instruction pointer (IP) in an interruption instruction pointer (IIP);

determine a first virtual address from an operating system mapping table;

determine a second virtual address from a translation look-aside buffer;

identify the errant process, if the physical address and the second virtual address match the physical address and the first virtual address.

21. The computer system of claim 20 further comprising:

an errant process physical address register, said errant process physical address register being configured to receive and store the physical address of the errant process that caused the error.

22. The computer system of claim 20 further comprising:

a mapping table, said mapping table being configured to maintain a mapping between all virtual addresses and all physical addresses in the computer system so that each virtual address is mapped to one physical address.

23. The computer system of claim 22, wherein the mapping table is implemented as one of:

a buffer; and

a cache array.

24. The computer system of claim 20 further comprising:

a translation look-aside buffer (TLB), said TLB being configured to store physical addresses.

25. The computer system of claim 24, the TLB further being configured to store second virtual addresses associated with the physical addresses.

26. The computer system of claim 24 further comprising:

a firmware component, said firmware component being configured to provide the physical addresses stored in the TLB.

27. The computer system of claim 25, said firmware component further configured to provide the second virtual addresses stored in the TLB.

28. The computer system of claim 20, said machine-readable medium comprising:

a non-volatile memory.